(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 November 2002 (28.11.2002)

PCT

(10) International Publication Number
WO 02/095588 A2

(51) International Patent Classification[7]: G06F 11/30

(21) International Application Number: PCT/US01/51581

(22) International Filing Date:
30 November 2001 (30.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/728,701 1 December 2000 (01.12.2000) US

(71) Applicant: NETWORK APPLIANCE, INC. [US/US]; 495 East Java Drive, Sunnyvale, CA 94089 (US).

(72) Inventor: MUHLESTEIN, Mark; 5831 E. Placita Alta Reposa, Tucson, AZ 85750 (US).

(74) Agent: SWERNOFSKY, Steven, A.; Swernofsky Law Group, P.O. Box 390013, Mountain View, CA 94039-0013 (US).

(81) Designated States *(national)*: CA, JP.

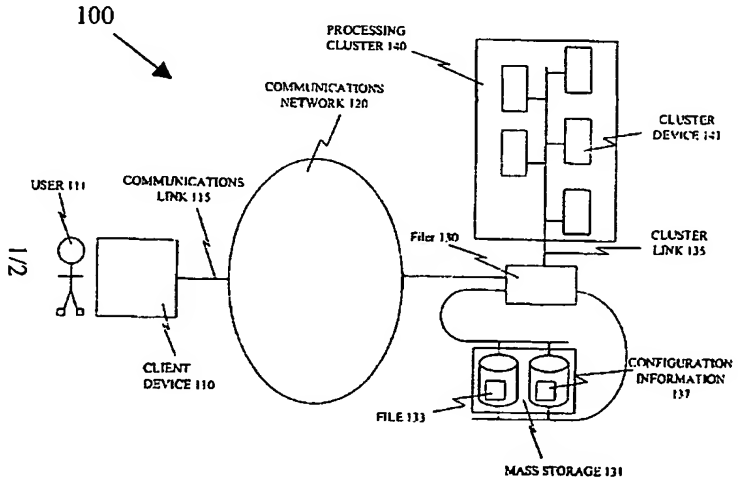
(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Declaration under Rule 4.17:
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: DECENTRALIZED VIRUS SCANNING FOR STORED DATA



(57) Abstract: The invention provides a method and system for performing specialized services for files at a server, such as scanning files for viruses. A filer or other server is connected to one or more supplementary computing devices that scan requested files to ensure they are virus free prior to delivery to end users. When an end user requests a file the following steps occur: The server determines whether the file requested must be scanned before delivery to the end user. The server opens a channel to one of the external computing devices and sends the filename. The external computing device opens the file and scans it. The external computing device notifies the filer the results of the file scan operation. The server sends the file to the end user provided the status indicates it may do so.

WO 02/095588 A2

DECENTRALIZED VIRUS SCANNING FOR STORED DATA

Background of the Invention

1. Field of the Invention

This invention relates to decentralized virus scanning for stored data, such as for example in a networked environment.

2. Related Art

Computer networking and the Internet in particular offer end users unprecedented access to information of all types on a global basis. Access to information can be as simple as connecting some type of computing device using a standard phone line to a network. With the proliferation of wireless communication, users can now access computer networks from practically anywhere.

Connectivity of this magnitude has magnified the impact of computer viruses. Viruses such as "Melissa" and "I love you" had a devastating impact on computer systems worldwide. Costs for dealing with viruses are often measured in millions and tens of millions of dollars. Recently it was shown that hand-held computing devices are also susceptible to viruses.

Virus protection software can be very effective in dealing with viruses, and virus protection software is widely available for general computing devices such as personal computers. There are, however, problems unique to specialized computing devices, such as such as for example servers, file servers, storage systems, and devices of any kind performing storage and retrieval of data. Off-the-shelf virus protection software will not run on a specialized computing device unless it is modified to do so, and it can be very expensive to rewrite software to work on another platform.

A first known method is to scan for viruses at the data source. When the data is being provided by a specialized computing device the specialized computing device must be scanned. Device-specific virus protection software must be written in order to scan the files on the device.

While this first known method is effective in scanning files for viruses, it suffers from several drawbacks. First, a company with a specialized computing device would have to dedicate considerable resources to creating virus protection software and maintaining up-to-date data files that protect against new viruses as they emerge.

Additionally, although a manufacturer of a specialized computing device could enlist the assistance of a company that creates mainstream virus protection software to write the custom application and become a licensee this would create other problems, such as reliance on the chosen vendor of the anti-virus software, compatibility issues when hardware upgrades are effected, and a large financial expense.

A second known method for protecting against computer viruses is to have the end user run anti-virus software on their client device. Anti-virus software packages are offered by such companies as McAfee and Symantec. These programs are loaded during the boot stage of a computer and work as a background job monitoring memory and files as they are opened and saved.

While this second known method is effective at intercepting and protecting the client device from infection, it suffers from several drawbacks. It places the burden of detection at the last possible link in the chain. If for any reason the virus is not detected prior to reaching the end user it is now at the computing device where it will do the most damage (corrupting files and spreading to other computer users and systems).

It is much better to sanitize a file at the source from where it may be delivered to millions of end users rather than deliver the file and hope that the end user is prepared to deal with the file in the event the file is infected. End users often have older versions of anti-virus software and/or have not updated the data files that ensure the software is able to protect against newly discovered viruses, thus making detection at the point of mass distribution even more critical.

Also, hand-held computing devices are susceptible to viruses, but they are poorly equipped to handle them. Generally, hand-held computing devices have very limited memory resources compared to desktop systems. Dedicating a portion of these resources to virus protection severely limits the ability of the hand-held device to perform effectively. Reliable virus scanning at the information source is the most efficient and effective method.

Protecting against viruses is a constant battle. New viruses are created everyday requiring virus protection software manufacturers to come up with new data files (solution algorithms used by anti-virus applications). By providing protection at the source of the file, viruses can be eliminated more efficiently and effectively.

Security of data in general is important. Equally important is the trust of the end user. This comes from the reputation that precedes a company, and companies that engage in web commerce often live and die by their reputation. Just like an end user trusts that the credit card number they have just disclosed for a web-based sales transaction is secure they want files they receive to be just as secure.

Accordingly, it would be desirable to provide a technique for scanning specialized computing devices for viruses and other malicious or unwanted content that may need to be changed, deleted, or otherwise modified.

Summary of the Invention

The invention provides a method and system for performing specialized services for files at a server, such as scanning files at a storage system, filer, or other server performing storage and retrieval of data, for viruses by secondary computing devices. The server (such as a filer) is connected to one or more supplementary computing devices that scan requested files upon request to ensure they are virus free prior to delivery to end users. When an end user requests a file from the server the following steps occur: The server determines whether the file or other object requested by the user must be scanned before delivery to, or after use by, the user. The server opens a channel to one of the external computing devices and sends the filename (or some other designator of the file or object, such as a file handle or an i-node pointer; "filename," "file name space" and the like refer to the collection of possible designators for files or other types of object). The external computing device opens the file and scans it. After possibly taking remedial actions (such as for example cleaning the file of the virus, quarantining or deleting the file), the external computing device notifies the filer the status of the file scan operation. The server sends the file to the end user provided the status indicates it may do so.

This system is very efficient and effective, as a file needs only to be scanned one time for a virus unless the file has been modified or new data files that protect against new viruses have been added. Scan reports for files that have been scanned may be stored in one or more of the external computing devices, in one or more servers, and some portion of a scan report may be delivered to end users.

In alternative embodiments of the invention one or more of the external computing devices may be running other supplementary applications, such as data compression and decompression, data encryption and decryption, and database compaction, independently or in some combination.

Brief Description of the Drawings

Figure 1 shows a block diagram of a system for decentralized appliance virus scanning.

Figure 2 shows a process flow diagram for a system for decentralized virus scanning

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. Those skilled in the art would recognize after perusal of this application that embodiments of the invention can be implemented using one or more general purpose processors or special purpose processors or other circuits adapted to particular process steps and data structures described herein, and that implementation of the process steps and data structures described herein would not require undue experimentation or further invention.

Lexicography

The following terms refer or relate to aspects of the invention as described below. The descriptions of general meanings of these terms are not intended to be limiting, only illustrative.

- **filer** — In general, this refers to any storage system, file server, or other device performing storage and retrieval of data. Storage systems might be implemented in any one of a large variety of ways, including but not limited to a network-attached storage environment; a storage area network; a disk assembly coupled to a client device, a server device, or a host computer; or some combination thereof.

One type of storage system is a file server. A file server or filer includes a computer that provides file services relating to the organization of information on writeable persistent storage devices, such as memories, tapes or disks of an array. The filer might include a storage operating system that implements a file system to logically organize the information as a hierarchical structure of directories and files on, e.g., the disks. Each "on-disk" file may be implemented as a set of data structures, e.g., disk blocks, configured to store information, such as the actual data for the file. A directory, on the other hand, might be implemented as a specially formatted file in which information about other files and directories are stored. In general, the term "storage operating system" refers to computer-executable code that implements data storage functionality, such as file system semantics, and manages data access. A storage operating system can be implemented as an application program operating over a general-purpose operating system, such as UNIX® or Windows NT®, or as a general-purpose operating system with storage functionality or with configurable functionality that is configured for storage applications, or as a special-purpose operating system dedicated to performing a limited range of functionality including storage and related tasks in storage appliances and other devices.

A storage system may be further configured to operate according to a client/server model of information delivery to thereby allow many clients to access files stored on a server, e.g., the storage system. In this model, the client may comprise an application executing on a computer that "connects" to the storage system over a computer network, such as a point-to-point link, shared local area network, wide area network or virtual private network implemented over a public network, such as the Internet. Each client may request the services of the file system on the storage system by issuing file system protocol messages (in the form of packets) to the system over the network. It should be noted, however, that the storage system may alternatively be configured to operate as an assembly of storage devices that is directly-attached to a (e.g., client or "host") computer. Here, a user

may request the services of the file system to access (i.e., read and/or write) data from/to the storage devices.

Although the invention is described herein with reference to a "filer," there is no particular limitation of the invention to filers, file servers, storage systems, or similar devices. It would be clear to those skilled in the art, after perusal of this application, how to implement the ideas and techniques described herein for all types of server devices. Such implementations would not require any undue experimentation or further invention, and are within the scope and spirit of the invention.

- **i-node** — In general, this refers to a directory entry or other file descriptor entry persistently maintained by a system performing storage and retrieval of data. In a preferred embodiment, each file has an i-node, and the i-node is persistently recorded in a directory for that file. Although the term "i-node" is sometimes referred to in the known art as being particular the Unix operating system and variants thereof, it is used in this description much more generally, as noted herein. There is no particular requirement in the invention that i-nodes must have any particular structure, or must be stored in any particular format or place, or are specific to any particular operating system, storage operating system, storage structure, hierarchical file system, file name space, or storage paradigm.

- **file** or other **object** — In general, this refers to any data object at the server, whether a sequential set of bytes, a set of records in a data base, a software object in an object-oriented database or an object-oriented language development environment, or any dynamically generated set of data for which a user request is appropriate. In a preferred embodiment, a file includes a set of data persistently recorded in a hierarchical namespace and having a set of file attributes. While this is preferred, there is no particular requirement that a file or other object requested by the user have these properties, or any particular other properties, as

the scope and spirit of the invention is broad enough to include all types of objects.

- **virus** — In general, this refers to any manmade program or piece of code that is loaded onto a computer without the computer user's knowledge and runs against their wishes. Most viruses can also replicate themselves, and the more dangerous types of viruses are capable of transmitting themselves across networks and bypassing security systems. A "virus" can also include any malicious code, program, or other internal component (including but not limited to a computer virus, computer worm, computer time bomb, Trojan horse, or component with similar effect), that could damage, destroy, alter, or take control of, software, firmware, or hardware, or could, in any manner, reveal, damage, destroy, or alter any data or other information accessed through or processed by the computer in any manner.

- **client** and **server** — in general, these terms refer to a relationship between two devices, particularly to their relationship as client and server, not necessarily to any particular physical devices.

  For example, but without limitation, a particular client device in a first relationship with a first server device, can serve as a server device in a second relationship with a second client device. In a preferred embodiment, there are generally a relatively small number of server devices servicing a relatively larger number of client devices.

- **client device** and **server device** — in general, these terms refer to devices taking on the role of a client device or a server device in a client-server relationship (such as an HTTP web client and web server). There is no particular requirement that any client devices or server devices must be individual physical devices. They can each be a single device, a set of cooperating devices, a portion of a device, or some combination thereof.

For example, but without limitation, the client device and the server device in a client-server relation can actually be the same physical device, with a first set of software elements serving to perform client functions and a second set of software elements serving to perform server functions.

Although the invention is described with regard to a client-server model, there is no particular requirement in the invention that the stored data is maintained and communicated to users using a client-server model. For example, other forms of distributed computing in which a user request for access to data objects triggers decentralized processing by one or more of a set of computing devices would also be within the scope and spirit of the invention.

As noted above, these descriptions of general meanings of these terms are not intended to be limiting, only illustrative. Other and further applications of the invention, including extensions of these terms and concepts, would be clear to those of ordinary skill in the art after perusing this application. These other and further applications are part of the scope and spirit of the invention, and would be clear to those of ordinary skill in the art, without further invention or undue experimentation.

System Elements

Figure 1 shows a block diagram of a system for decentralized appliance virus scanning.

A system 100 includes a client device 110 associated with a user 111, a communications network 120, a filer 130, and a processing cluster 140.

The client device 110 includes a processor, a main memory, and software for executing instructions (not shown, but understood by one skilled in the

art). Although the client device 110 and filer 130 are shown as separate devices there is no requirement that they be physically separate.

In a preferred embodiment, the communication network 120 includes the Internet. In alternative embodiments, the communication network 120 may include alternative forms of communication, such as an intranet, extranet, virtual private network, direct communication links, or some other combination or conjunction thereof.

A communications link 115 operates to couple the client device 110 to the communications network 120.

The filer 130 includes a processor, a main memory, software for executing instructions (not shown, but understood by one skilled in the art), and a mass storage 131. Although the client device 110 and filer 130 are shown as separate devices there is no requirement that they be separate devices. Moreover, although the invention is described with regard to a single filer 130, the invention is equally applicable to sets of filers 130 operating with the processing cluster 140. A set of multiple filers 130 might each one operate independently and each one make individual use of the processing cluster 140, or might operate in conjunction as a group and make use of the processing cluster 140 as a collective entity, or some combination thereof. Since, as noted below, the processing cluster 140 can include one or more cluster devices 141, the invention can be performed with any set of M filers and any set of N processors. There is no particular requirement that M or N must be fixed; either filers 130 or cluster devices 141 might be added by operator command or by a handshaking protocol while filers 130 and cluster devices 141 are operating. The filer 130 is connected to the communications network 120.

The filer 130 includes a set of configuration information 137 disposed so that a processor for the filer 130 can readily access that configuration information 137. In a preferred embodiment, the filer 130 includes software instructions for

reviewing, reporting, editing, or modifying the configuration information 137, as directed by an operator, or possibly by a remote user having designated privileges. The configuration information 137 includes the following:

- Information indicating a first set of file types for which virus scanning is enabled (such as executable files, often designated by the file name extension EXE), and a second set of file types for which virus scanning is disabled (such as raw text files, often designated by the file name extension TXT);

- Information indicating a first file space for which virus scanning is enabled for all file operations (such as a first CIFS "share" designated by its root directory, for example /users/Swernofsky), a second file space for which virus scanning is enabled for file write operations only (such as a second CIFS "share"), and a third file space for which virus scanning is disabled (again, such as a third CIFS "share");

and

- Information indicating for each file whether that file has been scanned for a virus, and if so, what date and time that scan was performed (such as a timestamp), by what type of scanning device or scanning software that scan was performed (such as the make and version number of the scanning software), and what the results of that scan were (such as whether a virus was detected and what actions were taken if a virus was in fact detected). In a preferred embodiment, this information is recorded in an i-node for the file, or if the file is read-only or if the i-node is unwritable (such as if the file is part of a read-only snapshot), in a separate scanning history database.

The mass storage 131 includes at least one file 133 that is capable of being requested by a client device 110. The processing cluster 140 includes one or more cluster device 141 each including a processor, a main memory, software for

executing instructions, and a mass storage (not shown but understood by one skilled in the art). Although the filer 130 and the processing cluster 140 are shown as separate devices there is no requirement that they be separate devices.

In a preferred embodiment the processing cluster 140 is a plurality of personal computers in an interconnected cluster capable of intercommunication and direct communication with the filer 130. There is no particular requirement that the processing cluster 140 must be organized as a unified cluster, or must be local to the filer 130, or must be homogeneous in the nature of the processing devices, or have any other particular characteristics. For example, in alternative embodiments, the processing cluster 140 includes a set of PC's, workstations, servers, or other devices, coupled to the filer 130 by means of a network such as the Internet.

In a preferred embodiment, cluster devices 141 in the processing cluster 140 register their presence with the filer 130, thus giving the filer 130 knowledge of their availability to perform scanning (or other) operations. While this is preferred, there is no particular requirement for the invention for registration, as the filer 130 may in alternative embodiments be configured to send out "John Doe" requests for cluster devices 141 to process files requested by the user.

The cluster link 135 operates to connect the processing cluster 140 to the filer 130. The cluster link 135 may include non-uniform memory access (NUMA), or communication via an intranet, extranet, virtual private network, direct communication links, or some other combination or conjunction thereof.

Method of Operation

Figure 2 shows a process flow diagram for a system for decentralized appliance virus scanning.

A method 200 includes a set of flow points and a set of steps. The system 100 performs the method 200. Although the method 200 is described serially, the steps of the method 200 can be performed by separate elements in conjunction or in parallel, whether asynchronously, in a pipelined manner, or otherwise. There is no particular requirement that the method 200 be performed in the same order in which this description lists the steps, except where so indicated.

At a flow point 210, the system 100 is ready to begin performing the method 200.

At a step 211, a user 111 utilizes the client device 110 to initiate a request for a file 133. The request is transmitted to the filer 130 via the communications network 120. In a preferred embodiment the filer 130 is an independent file server performing file retrieval and storage in response to a file server protocol such as NFS or CIFS. In alternative embodiments, the filer 130 might be a supplemental storage device or file maintenance server operating at the direction of another server, such as a web server.

At a step 212, the filer 130 receives the request for the file 133 and determines if the file 133 must be scanned for a virus. As part of this step, the filer 130 performs the following sub-steps:

- At a sub-step 212(a), the filer 130 reviews its information regarding whether the file 133 has already been scanned for a virus. In a preferred embodiment, that information includes whether a scan has already been performed, what date and time that scan was performed (such as a timestamp), by what type of scanning device or scanning software that scan was performed (such as the make and version number of the scanning software), and what the results of that scan were (such as whether a virus was detected and what actions were taken if a virus was in fact detected). As noted above, in a preferred embodiment, this information is recorded in the i-node for the file 133. If the

file 133 has already been scanned and is marked available for use (and the filer determines that no re-scan is required), the filer 130 makes the file available to the user without performing the scanning operation.

- At a sub-step 212(b), the filer 130 reviews its information regarding what types of files 133 it should scan for a virus. The filer reviews its configuration information 137 describing a set of file types (1) that should be scanned for a virus, such as executable files, macros, scripts, and the like, and (2) that should not be scanned for a virus, such as raw text files and the like. This set of file types might be selected by an operator for the filer 130, and is maintained with the configuration information 137. In a preferred embodiment, file types are identified by portions of the file name for the file 133, such as a file name extension. Known file name extensions include EXE for executable files and TXT for raw text files.

- At a sub-step 212(c), the filer 130 reviews its information regarding what file spaces it should scan for a virus. The filer reviews its configuration information 137 describing which file spaces should be scanned for (1) all file operations, (2) only file write operations, or (3) no file operations. Where the file space should be scanned for all file operations, the filer 130 causes the file 133 to be scanned before the file 133 is opened for any read operation and after the file 133 is closed after a write operation. Where the file space should be scanned for only file write operations, the filer 130 causes the file 133 to be scanned after the file 133 is closed after a write operation.

At a step 213, the filer 130, having determined that the file 133 should be scanned, sends the file ID and path of the file 133 to the processing cluster 140 where it is received by one of the cluster devices 141. As part of this step, the filer 130 performs the following sub-steps:

- At a sub-step 213(a), the filer 130 sets a timer to a cluster processor timeout value, indicating how long the filer 130 is willing to wait for a cluster device 141 to work.

- At a sub-step 213(b), the filer 130 waits for the cluster device 141 to complete its work. While doing so, the cluster device 141 (hopefully) performs step 215, step 217, and step 219 described below.

- At a sub-step 213(c), if the cluster device 141 responds before the timeout, the filer 130 proceeds with the step 219 below, using the results from the cluster device 141.

- At a sub-step 213(d), if the cluster device 141 does not respond before the timeout, the filer 130 might proceed in one of two ways: (a) The filer 130 proceeds with the step 219 below, acting as if the cluster device 141 refused user access to the file. In this case, the filer 130 reports that the file is not available due to the scan having failed. (b) The filer 130 sends an ARE-YOU-WORKING? message to the cluster device 141. In this case, if the cluster device 141 responds, within a second but shorter timeout, that it is still working on the file 133, the filer 130 returns to the sub-step 213(b) and resets the timeout.

In a preferred embodiment, there is more than one cluster device 141, so the filer 130 can proceed to service requests for other files 133 even if the cluster device 141 scanning one particular file 133 takes a very long time. In alternative embodiments, the filer 130 may reassign the scanning task to a second cluster device 141 if the filer 130 suspects that the first cluster device 141 has in fact crashed, become unavailable, or otherwise is not likely to respond successfully with a virus scan result for the file 133.

- In the event that the user making the original request for the file 133 gives up before the cluster device 141 reports on the file 133, the filer 130 still waits for the report from the cluster device 141, and marks the file 133 with the results of the virus scan performed by the cluster device 141. Thus, if the cluster device 141 determines that the file 133 has no virus (or alternatively, finds a virus but successfully removes it), the filer 130 marks the file as successfully scanned and available for use. If the same user or a different user later requests the same file 133, the filer 130 makes that file 133 available without a further scan, as described below.

At a step 215, the cluster device 141 uses the file ID and path to open the file 133 in the mass storage 131 of the filer 130.

At a step 217, the cluster device 141 scans the file 133 for viruses. In a preferred embodiment, files are tasked to the processing cluster 140 in a round robin fashion. In alternative embodiments files may be processed individually by a cluster device 141, by multiple cluster device 141 simultaneously, or some combination thereof. Load balancing may be used to ensure maximum efficiency of processing within the processing cluster 140.

In a preferred embodiment, the filer 130 groups cluster devices 141 into one or more classes, such as primary and secondary, where all primary cluster devices 141 are assigned, followed by secondary cluster devices 141. This allows an operator to direct the filer 130 to use a first cluster device 141, such as for example available using a relatively rapid connection, exclusively, but when the first cluster device 141 is unavailable for any reason, to fall back to using a second designated cluster device 141, such as for example available using a much less rapid connection.

There are several vendors offering virus protection software for personal computers, thus the operator of the filer 130 may choose whatever product they would like to use that supports the communication protocol with the filer 130

described herein. They may even use combinations of vendors' products in the processing cluster 140, when those combinations can operate using the communication protocol with the filer 130 described herein. In alternative embodiments, the filer 130 may operate with forms of virus protection software that does not support the communication protocol with the filer 130 described herein, with some features (such as the timeout and ARE-YOU-WORKING? message) not available to those forms of virus protection software. In further alternative embodiments of the invention, continual scanning of every file 133 on the filer 130 may take place.

The processing cluster 140 is highly scalable. The price of personal computers is low compared to dedicated devices, such as filers, therefore this configuration is very desirable. Additionally, a cluster configuration offers redundant systems availability in case a cluster device 141 fails – failover and takeover is also possible within the processing cluster.

The cluster device 141 is assigned a special type of access (herein called "OPEN-FOR-SCANNING"), so that the cluster device 141 can scan the file 133 regardless of whether it is already locked by another user. In a preferred embodiment, OPEN-FOR-SCANNING mode is restricted to those devices the filer 130 can verify are actually cluster devices 141. In a preferred embodiment, the filer 130 can restrict OPEN-FOR-SCANNING mode to devices according to one or more of the following criteria:

- having one or more selected IP addresses;
- being included in one or more selected IP subnets;
- being included in one or more selected DNS domains;
- being accessible to the filer 130 via one or more selected physical interfaces;

- having a selected username or user privileges (such as "Administrator" or "Backup Operator") at the cluster device 141.

In a preferred embodiment, OPEN-FOR-SCANNING mode access is restricted to processes running as an NT "Service" on the cluster device 141. Thus, a selected cluster device 141 might be in use by a user having no particularly special privileges, while the cluster device 141 concurrently operates with a service running as "Administrator" and thus being allowed by the filer 130 to have OPEN-FOR-SCANNING mode access.

At a step 219, the cluster device 141 transmits a scan report to the filer 130. The scan report primarily reports whether the file is safe to send. Further information may be saved for statistical purposes (for example, how many files have been identified as infected, was the virus software able to sanitize the file or was the file deleted) to a database. The database may be consulted to determine whether the file 133 needs to be scanned before delivery upon receipt of a subsequent request. If the file 133 has not changed since it was last scanned and no additional virus data files have been added to the processing cluster, the file 133 probably does not need to be scanned. This means the file 133 can be delivered more quickly.

Other intermediary applications may also run separately, in conjunction with other applications, or in some combination thereof within the processing cluster 140. Compression and encryption utilities are some examples of these applications. These types of applications, including virus scanning, can be very CPU intensive, thus outsourcing can yield better performance by allowing a dedicated device like a filer to do what it does best and farm out other tasks to the processing cluster 140.

As part of this step, the filer 130 might also perform the following sub-steps:

- At a sub-step 219(a), the filer 130 records information from the scan report in the i-node for the file 133, or in a separate scanning history database if the file is read-only, as noted above.

- In a preferred embodiment, the filer 130 includes software instructions for responding to an operator or a privileged remote user to reset the scanning information for a file. This allows an operator or a privileged remote user to force the filer 130 to rescan one or more selected files 133.

At a step 221, the filer 130 transmits or does not transmit the file 133 to the client 110 based on its availability as reported following the scan by the processing cluster 140. Some portion of the scan report may also be transmitted to the user. As part of this step, the filer 130 performs the following sub-steps:

- At a sub-step 221(a), if the report from the cluster device 141 indicates that the file 133 is unavailable due to being infected (and the file 133 was not disinfected by the cluster device 141), the filer 130 sends a message box to the requesting user giving at least some information from the report from the cluster device 141. The filer 130 can send this message box to a user making a CIFS request because the CIFS protocol allows the filer 130 to know the IP (internet protocol) address for the user. For NFS, the filer 130 would build a string indicating a path to the requested file 133 and send a message to the user including that string.

At this step, a request for a file 133 has been received, the request has been processed, and if possible a file 133 has been delivered. The process may be repeated at step 211 for subsequent requests.

Generality of the Invention

The invention has wide applicability and generality to other aspects of processing requests for files.

The invention is applicable to one or more of, or some combination of, circumstances such as those involving:

- file compression and decompression — the cluster processors can be used to decompress data for delivery to users, and to compress data received from users for storage.

- file encryption and decryption — the cluster processors can be used to decrypt data for delivery to users, and to encrypt data received from users for storage.

- database compaction — the cluster processors can be used to compact data in a database or other structured format for delivery to users, or to compact data received from users for storage.

- general outsourcing of CPU intensive tasks from dedicated appliances to general purpose computers — for one example, the cluster processors can be used to translate between data stored in a first form into data presented to users in a second form.

Alternative Embodiments

Although preferred embodiments are disclosed herein, many variations are possible which remain within the concept, scope, and spirit of the invention, and these variations would become clear to those skilled in the art after perusal of this application.

Claims

1. A method including

receiving a user request for an object at a server;

performing an operation on data associated with said object at a cluster device, said operation including accessing said object at said server; and

conditionally allowing access to said object in response to said user request and a result of said operation.

2. A method as in claim 1, including conditioning said operation on a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share.

3. A method as in claim 1, including conditioning said operation on an intersection of

a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share; and

a type of access associated with said user request;

wherein said operation is performed for an intersection of at least one said feature and at least one type of access.

4. A method as in claim 1, including persistently recording a result of said operation in association with said object.

5. A method as in claim 1, including selecting said cluster device to perform said operation in response to a priority class associated with said cluster device.

6. A method as in claim 1, wherein said operation includes a plurality of processes, each one process being performed at a separate cluster device.

7. A method as in claim 1, wherein said operation includes at least one of: virus scanning, encryption or decryption, compression or decompression.

8. A method as in claim 1, wherein said operation includes

setting a timeout at said server;

resetting said timeout in response to receiving a response from said cluster device to a protocol message asking if said cluster device is still working on said operation; and

determining that said operation is successful in response to receiving a response from said cluster device before said timeout expires.

9. A method as in claim 1, including assigning an access type to said cluster device, said access type allowing said cluster device to perform said operation notwithstanding user locks associated with said object.

10. A method as in claim 9, including restricting said access type in response to at least one of: a selected set of network addresses for said cluster device, a selected set of domain names for said cluster device, a selected set of user names at said cluster device, a selected set of interfaces between said server and said cluster device.

11. A method as in claim 1, including

at a first time, recording a result of said operation for said object; and

at a second time, conditioning said operation on said result.

12. A method as in claim 11, wherein said result includes at least one of: a time when said operation was performed, remedial measures taken in response to said operation, whether access was allowed in response to said operation.

13. A method as in claim 1, including conditioning said operation on a type of access associated with said user request.

14. A method as in claim 13, wherein said operation is performed before allowing access to said object for requests including read access.

15. A method as in claim 13, wherein said operation is performed after allowing access to said object for requests including write access.

16. Apparatus including

a server having a set of objects and a network interface;

a user request for at least one requested one of said objects;

a cluster device;

a first message from said server to said cluster device, said first message indicating said requested one object;

a second message from said cluster device to said server, said second message indicating a result of an operation performed on said requested one object; and

a response to said user request, said response including conditional access to said object in response to said second message.

17. Apparatus as in claim 16, wherein said first message is responsive to a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share.

18. A method as in claim 16, wherein said first message is responsive to an intersection of

a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share; and

a type of access associated with said user request.

19. Apparatus as in claim 16, wherein said first message is directed at a selected said cluster device in response to a priority class associated with said cluster device.

20. Apparatus as in claim 16, including a plurality of said first messages directed at separate said cluster devices in response to a single said user request.

21. Apparatus as in claim 16, wherein said second message includes a result of at least one of: virus scanning, encryption or decryption, compression or decompression.

22. Apparatus as in claim 16, including a persistent record of at least some information responsive to said second message, said persistent record being associated with said object.

23. Apparatus as in claim 22, wherein said persistent record includes at least one of: a time when said second message was received, remedial measures taken by said cluster device in response to said first message, whether access was allowed in response to said user request.

24. Apparatus as in claim 16, wherein said conditional access is responsive to a type of access associated with said user request.

25. Apparatus as in claim 24, wherein said second message is received before allowing access to said object for user requests including read access.

26. Apparatus as in claim 24, wherein said first message is sent after allowing access to said object for user requests including write access.

27. Memory or mass storage including instructions interpretable by a computing device, said instructions directing said computing device to

receive a user request for an object at a server;

perform an operation on data associated with said object at a cluster device, said operation including accessing said object at said server; and

conditionally allow access to said object in response to said user request and a result of said operation.

28. Memory or mass storage as in claim 27, including instructions directing said computing device to condition said operation on a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share.

29. Memory or mass storage as in claim 27, including instructions directing said computing device to condition said operation on an intersection of

a feature of said object, said feature including at least one of: a file name, a file type, a filesystem share; and

a type of access associated with said user request;

wherein said operation is performed for an intersection of at least one said feature and at least one type of access.

30. Memory or mass storage as in claim 27, including instructions directing said computing device to persistently record a result of said operation in association with said object.

31. Memory or mass storage as in claim 27, including instructions directing said computing device to select said cluster device to perform said operation in response to a priority class associated with said cluster device.

32. Memory or mass storage as in claim 27, wherein said operation includes a plurality of processes, each one process being performed at a separate cluster device.

33. Memory or mass storage as in claim 27, wherein said operation includes at least one of: virus scanning, encryption or decryption, compression or decompression.

34. Memory or mass storage as in claim 27, wherein said operation includes

setting a timeout at said server;

resetting said timeout in response to receiving a response from said cluster device to a protocol message asking if said cluster device is still working on said operation; and

determining that said operation is successful in response to receiving a response from said cluster device before said timeout expires.

35. Memory or mass storage as in claim 27, including instructions directing said computing device to assign ing an access type to said cluster device, said access type allowing said cluster device to perform said operation notwithstanding user locks associated with said object.

36. Memory or mass storage as in claim 35, including instructions directing said computing device to restrict said access type in response to at least one of: a selected set of network addresses for said cluster device, a selected set of domain names for said cluster device, a selected set of user names at said cluster device, a selected set of interfaces between said server and said cluster device.

37. Memory or mass storage as in claim 27, including instructions directing said computing device to

at a first time, record a result of said operation for said object; and

at a second time, condition said operation on said result.

38. Memory or mass storage as in claim 37, wherein said result includes at least one of: a time when said operation was performed, remedial measures taken in response to said operation, whether access was allowed in response to said operation.

39. Memory or mass storage as in claim 27, including instructions directing said computing device to condition said operation on a type of access associated with said user request.

40. Memory or mass storage as in claim 39, wherein said operation is performed before allowing access to said object for requests including read access.

41. Memory or mass storage as in claim 39, wherein said operation is performed after allowing access to said object for requests including write access.

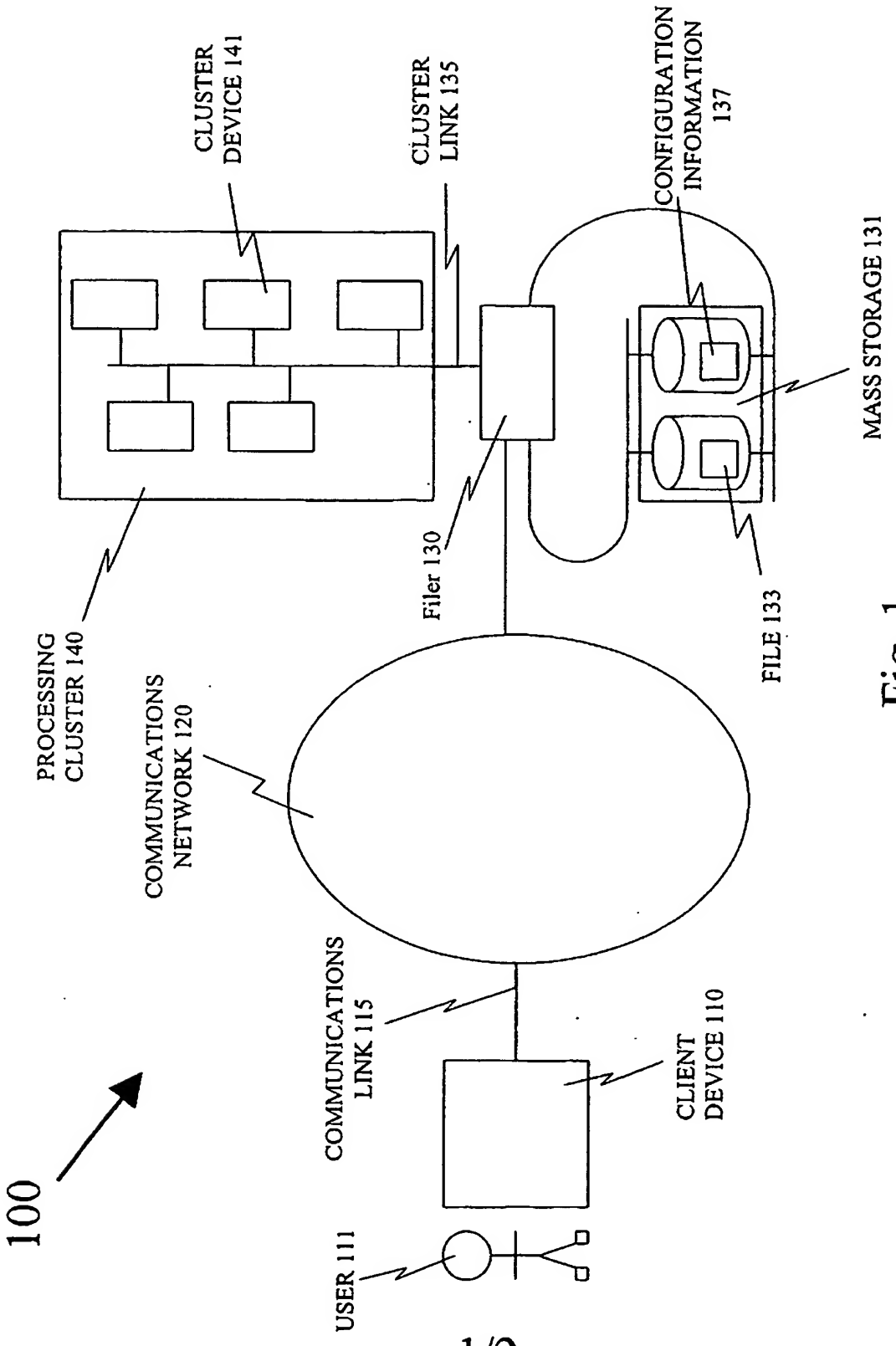

100
CLUSTER DEVICE 141
CLUSTER LINK 135
CONFIGURATION INFORMATION 137
MASS STORAGE 131
PROCESSING CLUSTER 140
Filer 130
FILE 133
COMMUNICATIONS NETWORK 120
COMMUNICATIONS LINK 115
CLIENT DEVICE 110
USER 111

Fig. 1

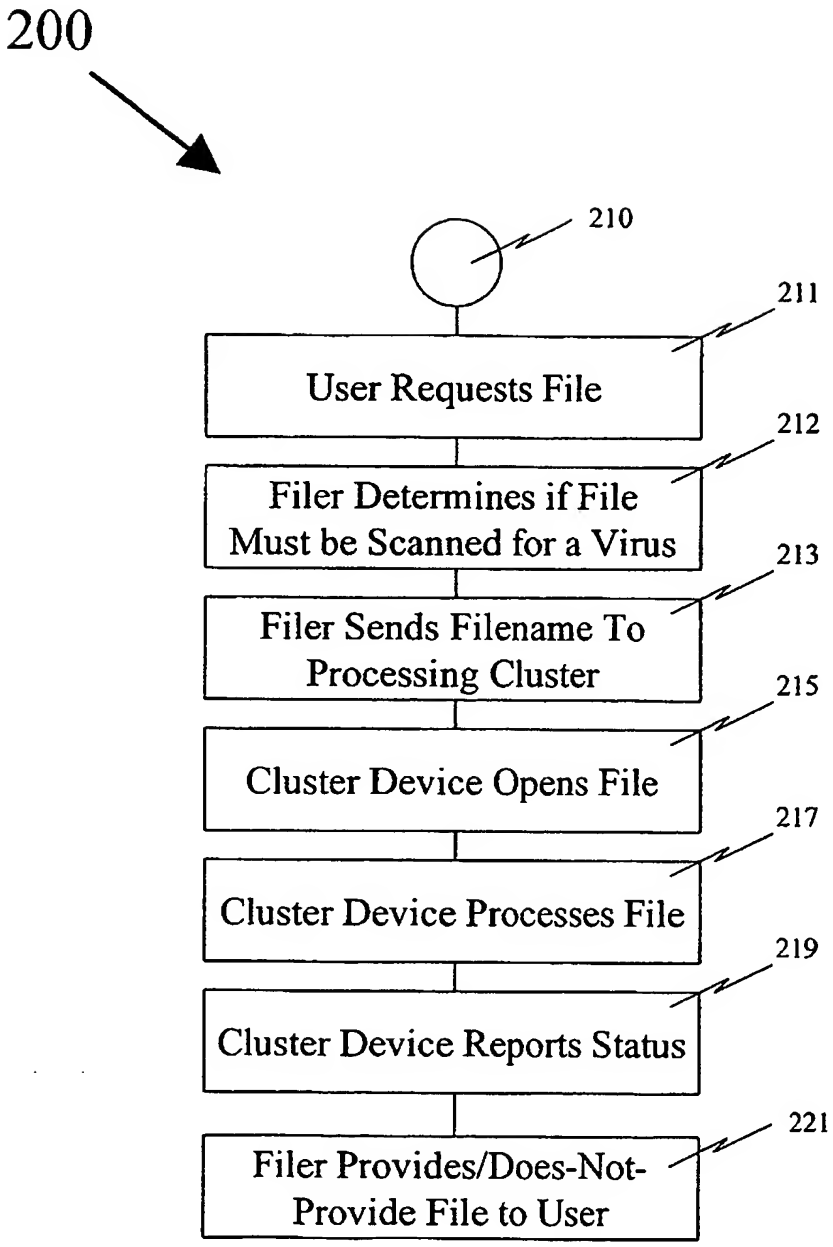

200
210
211
User Requests File
212
Filer Determines if File Must be Scanned for a Virus
213
Filer Sends Filename To Processing Cluster
215
Cluster Device Opens File
217
Cluster Device Processes File
219
Cluster Device Reports Status
221
Filer Provides/Does-Not-Provide File to User

Fig. 2